## CLAIMS

Amend the claims as follows.

1. (Currently amended) A method comprising:

detecting an interrupt service request;

inserting interrupt servicing instructions into an instruction queue mechanism in response to detecting the interrupt service request where inserting the interrupt servicing instructions into the instruction queue mechanism results in processor bandwidth being allocated between the inserted interrupt servicing instructions and other program instructions via concurrent in-line ~~stating~~ staging of the interrupt servicing instructions and other program instructions; and

executing the interrupt servicing instructions and other program instructions.

2. (Previously presented) The method of claim 1 where the instruction queue mechanism includes an instruction cache and an instruction fetch unit to fetch instructions from the instruction cache, the processing being performed in such manner as to decode the instructions into micro-opcodes and execute the micro-opcodes in one or more out-of-order execution units.

3. (Previously presented) The method of claim 2 comprising recycling the executed micro-opcodes for optional re-execution thereof in the one or more out-of-order execution units by retiring the executed micro-opcodes including those micro-opcodes representing the inserted interrupt servicing instructions to the instruction cache.

4. (Previously presented) The method of claim 3 where the executed micro-opcodes are retired to the instruction cache in order.

5. (Previously presented) The method of claim 1 where detecting comprises detecting plural interrupts by prioritizing the plural interrupts and inserting one or more instances of the interrupt servicing instructions into the instruction queue mechanism in accordance with one or more predefined interrupt servicing allocation criteria.

6. (Previously presented) The method of claim 5 where the one or more allocation criteria include the priority of the interrupts and the capacity of the processor to allocate bandwidth to interrupt servicing.

7. (Previously presented) The method of claim 6 where the prioritizing is dynamically responsive to changing allocation criteria.

8. (Previously presented) The method of claim 1 where the processor bandwidth is allocated to interrupt servicing without flushing the instruction queue mechanism.

9. (Previously presented) The method of claim 1 where the detecting is performed by an interrupt processor, which after the detecting, the method comprising:

at the interrupt processor, determining whether the detected interrupt service request is of a priority meeting one or more defined high-priority criteria and if so then signaling the processor to perform the inserting; and

at the core processor, responding to said signaling by performing said inserting and said processing.

10. (Currently amended) A ~~The~~ method ~~of claim 1 where the detecting is performed by an interrupt processor, the method~~ comprising:

detecting an interrupt service request;

inserting interrupt servicing instructions into an instruction queue mechanism in response to detecting the interrupt service request where inserting the interrupt servicing instructions into the instruction queue mechanism results in processor bandwidth being allocated between the inserted interrupt servicing instructions and other program instructions via concurrent in-line staging of the interrupt servicing instructions and other program instructions;

executing the interrupt servicing instructions and other program instructions;

determining that a natural processor context switch is imminent, said determining including analyzing one or more instructions within the instruction queue mechanism for context switch-inducing instructions;

signaling the interrupt processor to make ready the highest priority interrupt service request; and

signaling the instruction queue mechanism to fetch the readied interrupt service request in advance of a context switch responsive to the determining;

where the detecting is performed by an interrupt processor.

11. (Previously presented) A processor, comprising:

an instruction cache;

a fetch unit to fetch instructions from the instruction cache;

a decode unit to decode the instructions into micro-opcodes;

a dispatch unit to schedule the micro-opcodes for execution;

an execute unit to execute the micro-opcodes; and

an interrupt-handler to signal the fetch and decode units such that micro-opcodes corresponding to interrupt servicing instructions are inserted into an instruction sequence within the instruction cache without flushing the instruction cache.

12. (Currently amended) The processor of claim 11

where the computer supports plural hardware interrupt inputs;

where the interrupt-handler includes an interrupt concentrator to determine priority among the plural hardware interrupt inputs and to schedule the plural hardware interrupt inputs; and

where the interrupt-handler instructs the fetch and decode units such that plural instances of such decoded micro-opcodes are inserted, each instance representing one or more corresponding sets of interrupt servicing instructions, into the normal instruction sequence for serial scheduling and execution of the plural instances of such decoded micro-opcodes by the dispatch and execute units in accordance with the determined priority of the plural hardware interrupt inputs.

13. (Currently amended) A ~~The~~ processor ~~of claim 11~~ comprising:

an instruction cache;

a fetch unit to fetch instructions from the instruction cache;

a decode unit to decode the instructions into micro-opcodes;

a dispatch unit to schedule the micro-opcodes for execution;

an execute unit to execute the micro-opcodes;

an interrupt-handler to signal the fetch and decode units such that micro-opcodes corresponding to interrupt servicing instructions are inserted into an instruction sequence within the instruction cache without flushing the instruction cache; and

a context switch prediction unit coupled with the fetch and decode units to predict a naturally occurring context switch and to signal the interrupt-handler upon such prediction,

said context switch prediction unit to analyze one or more instructions within an instruction cache for context switch-inducing instructions~~,~~;

where the ~~said~~ interrupt-handler mechanism coupled to such signaling from said prediction unit instructing the fetch and decode units.

14. (Currently amended) An apparatus, comprising:

an instruction queue mechanism to stage p-code for execution;

a plurality of interrupt inputs corresponding to a plurality of hardware input/output devices;

interrupt priority logic to prioritize a plural interrupt corresponding to the plurality of interrupt inputs;

bandwidth allocation logic to allocate processing resources to service the plurality of interrupt according to their relative priority;

a p-code insertion mechanism to insert interrupt servicing p-code ~~in~~ into the instruction queue mechanism according to the processing resources allocated to the plurality of interrupt inputs;

a p-code processor coupled to the instruction queue mechanism and to the insertion mechanism to execute the interrupt servicing p-code

~~an~~ retirement unit to retire executed instructions back to said instruction cache.

15. (Previously presented) The apparatus of claim 14,

where the instruction queue mechanism includes an instruction cache and an in-order instruction unit to perform branch predictions; and

where the p-code processor includes a dispatch and out-of-order execution unit to schedule and execute p-code in parallel among one or more execution ports.

16. (Cancelled)

17. (Previously presented) The apparatus of claim 15 where the in-order instruction unit includes said allocation logic.

18. (Previously presented) The apparatus of claim 17 where the allocation logic is coupled to a computer operating system.

19. (Previously presented) The apparatus of claim 14 where the priority logic is coupled to a computer operating system.

20. (Previously presented) The apparatus of claim 14 where the allocation logic is coupled to a current-usage model.

21. (Previously presented) The apparatus of claim 14 where the priority logic is coupled to a current-usage model.

22. (Previously presented) A machine-readable medium comprising a computer-readable medium containing instructions, that, when executed by a machine cause the machine to perform a method comprising:

detecting an interrupt service request;

inserting interrupt servicing instructions responsive to the interrupt service request into an instruction queue mechanism;

processing the instructions within the instruction queue mechanism including the inserted interrupt servicing instructions; and

signaling after detecting an impending natural context switch represented by a recognized instruction sequence within the instruction queue mechanism;

where detecting an impending natural context switch is predicated on an instruction stream within the instruction queue mechanism of the machine.

23. (Previously presented) The article of claim 22 comprising:

signaling with a vector representing an interrupt service request upon determining that an interrupt service request of a priority that exceeds a given threshold has occurred; and

fetching interrupt service instructions from memory in accordance with the vector representing the determined priority interrupt service request.

24. (Cancelled)

25. (Currently amended) A computer system, comprising:

one or more input/output (I/O) devices to generate one or more hardware interrupts;

an interrupt concentrator to rank and queue the interrupts into an ordered interrupt array;

a core processor coupled to the interrupt concentrator, the core processor including an instruction cache, fetch unit, and a decode unit, the fetch unit to fetch instructions from the instruction cache and the decode unit to decode the interrupt service instructions and other program instructions into micro-opcodes, the core processor including a dispatch unit to schedule the micro-opcodes and an execute unit having one or more execution ports to execute the micro-opcodes in the one or more execution ports and thereafter retire the micro-opcodes back into the instruction cache; and

an interrupt-handler responsive to the one or more hardware interrupts, the interrupt-handler instructing the fetch and decode units to signal insertion into an instruction sequence decoded micro-opcodes representing interrupt servicing instructions to schedule and execute by the dispatch and execute units.

26. (Previously presented) The computer system of claim 25 which supports plural hardware interrupt inputs, where the interrupt-handler includes an interrupt concentrator to determine priority among and to schedule the plural hardware interrupts, the interrupt-handler instructing the fetch and decode units to insert plural instances of such decoded micro-opcodes, each instance representing one or more corresponding sets of interrupt servicing instructions, into the normal instruction sequence for serial scheduling and execution of the plural instances of such decoded micro-opcodes by the dispatch and execute units in accordance with the determined priority of said plural hardware interrupts.

27. (Previously presented) The computer system of claim 25 comprising:

a context switch prediction unit coupled with the fetch and decode units to predict a naturally occurring context switch represented by a recognized instruction sequence within the instruction cache and to signal the interrupt-handling mechanism upon such prediction;

where the interrupt-handling mechanism is coupled to such signaling from the prediction unit instructing the fetch and decode units.

28. (Currently amended) A method, comprising:

detecting an interrupt service request;

inserting interrupt servicing instructions responsive to the interrupt service request into an instruction queue mechanism in such manner that the inserted interrupt servicing instructions intervene mainline program instructions within a queue thereby allocating core

processor bandwidth between the inserted interrupt servicing instructions and the mainline program instructions;

processing the instructions within the instruction queue mechanism including the mainline program instructions and the inserted interrupt servicing instructions, where core processor bandwidth allocation between the mainline program instructions and the inserted interrupt servicing instructions is achieved by concurrent in-line staging of the same within the instruction queue mechanism without first flushing the instruction queue mechanism of its contents; and

recycling ~~the~~ executed micro-opcodes for optional re-execution thereof in the one or more out-of-order execution units by reordering and retiring the executed micro-opcodes including those micro-opcodes representing the inserted interrupt servicing instructions to the instruction cache.